Leveraging Automation in the Testing of Autonomous Spacecraft Systems

Martin S. Feather & Ben Smith
(Quality Assurance Office) (Information and Computing Technologies Research Section)
Jet Propulsion Laboratory
California Institute of Technology

Roadmap

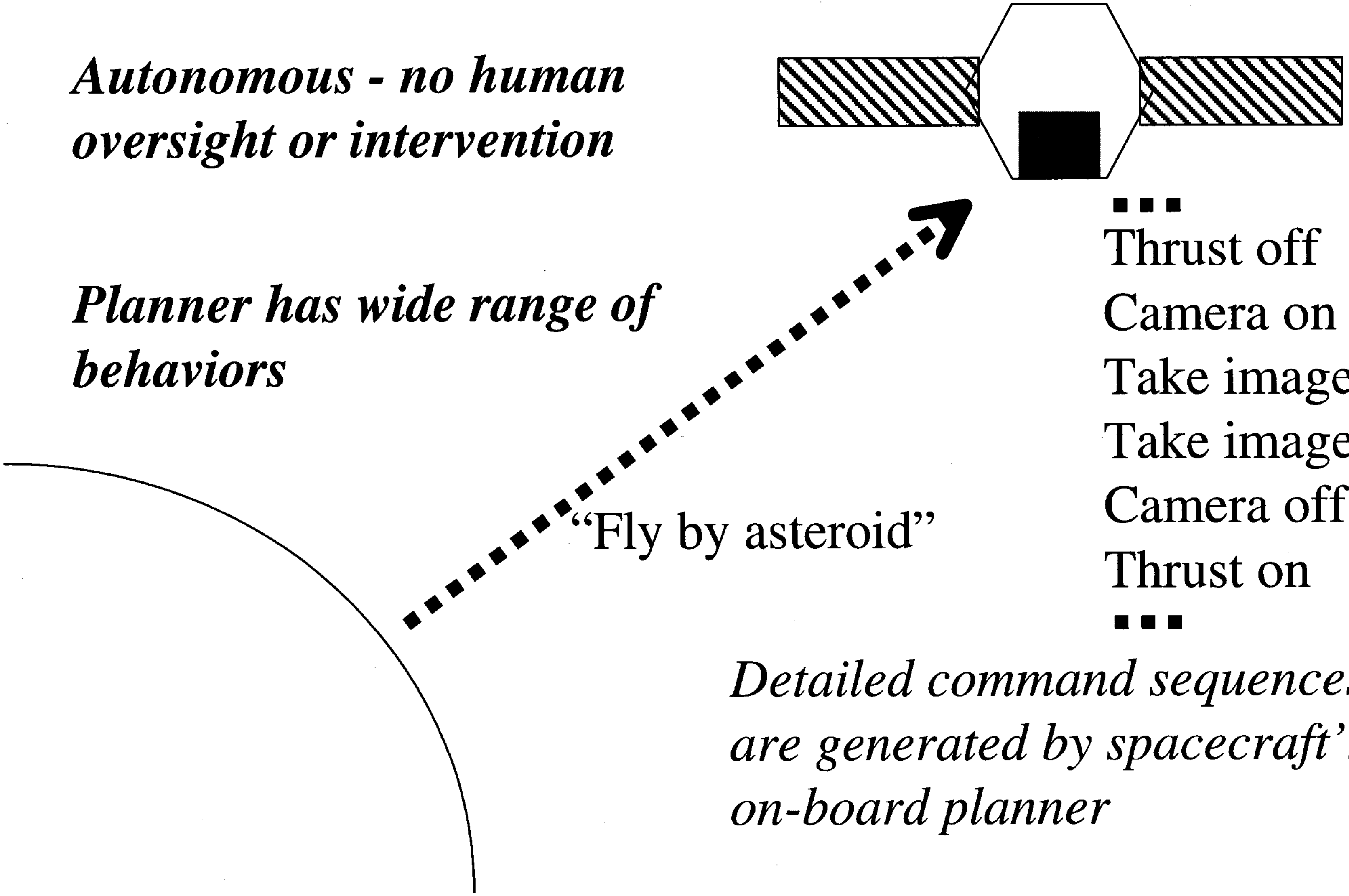
- Spacecraft's autonomous planner
- Testing challenges
- Example fragments
- Automate checking of flight rules
- Metrics
- Redundancy & Rationale
- Validation
- Applicability - worthwhile & viable
- Partnership development

Sponsors: NASA's Software Program (administered by the NASA IV&V Facility in Fairmont, WV) with funding provided by NASA's Office of Safety and Mission Assurance (Code Q) UPN #323-08-5b, and by the Autonomy Technology Program within the DS-1 project.

Spacecraft's autonomous planner

Autonomous - no human oversight or intervention

Planner has wide range of behaviors



"Fly by asteroid"

■ ■ ■

Thrust off

Camera on

Take image

Take image

Camera off

Thrust on

■ ■ ■

Detailed command sequences are generated by spacecraft's on-board planner

Some testing challenges

- Each plan must satisfy every one of the 200+ flight rules

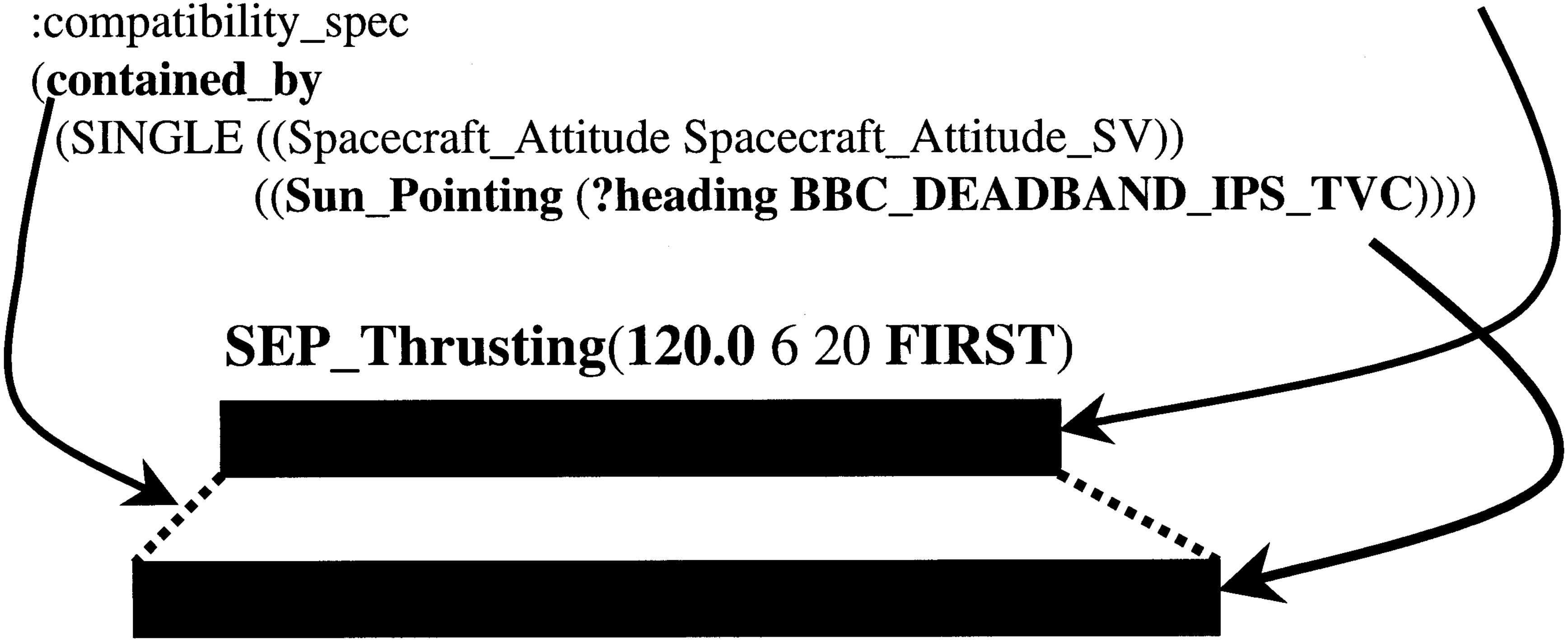
  each flight rule is a temporal relationship between activities

  e.g., thrusting activity *contained-by* constant-pointing-on-sun
- Plans are detailed and voluminous (1,000 - 5,000 lines)
- Information dispersed throughout plan
- Thorough testing yields thousands of plans

Manual inspection impractical - need automation!

Example flight rule

```
(Define_Compatibility
        (SINGLE ((SEP SEP_SV))
                ((SEP_Thrusting (?heading ?level ?duration FIRST))))
 :compatibility_spec
 (contained_by
  (SINGLE ((Spacecraft_Attitude Spacecraft_Attitude_SV))
          ((Sun_Pointing (?heading BBC_DEADBAND_IPS_TVC))))
```



SEP_Thrusting(120.0 6 20 **FIRST)**

Sun_Pointing (120.0 BBC_DEADBAND_IPS_TVC)

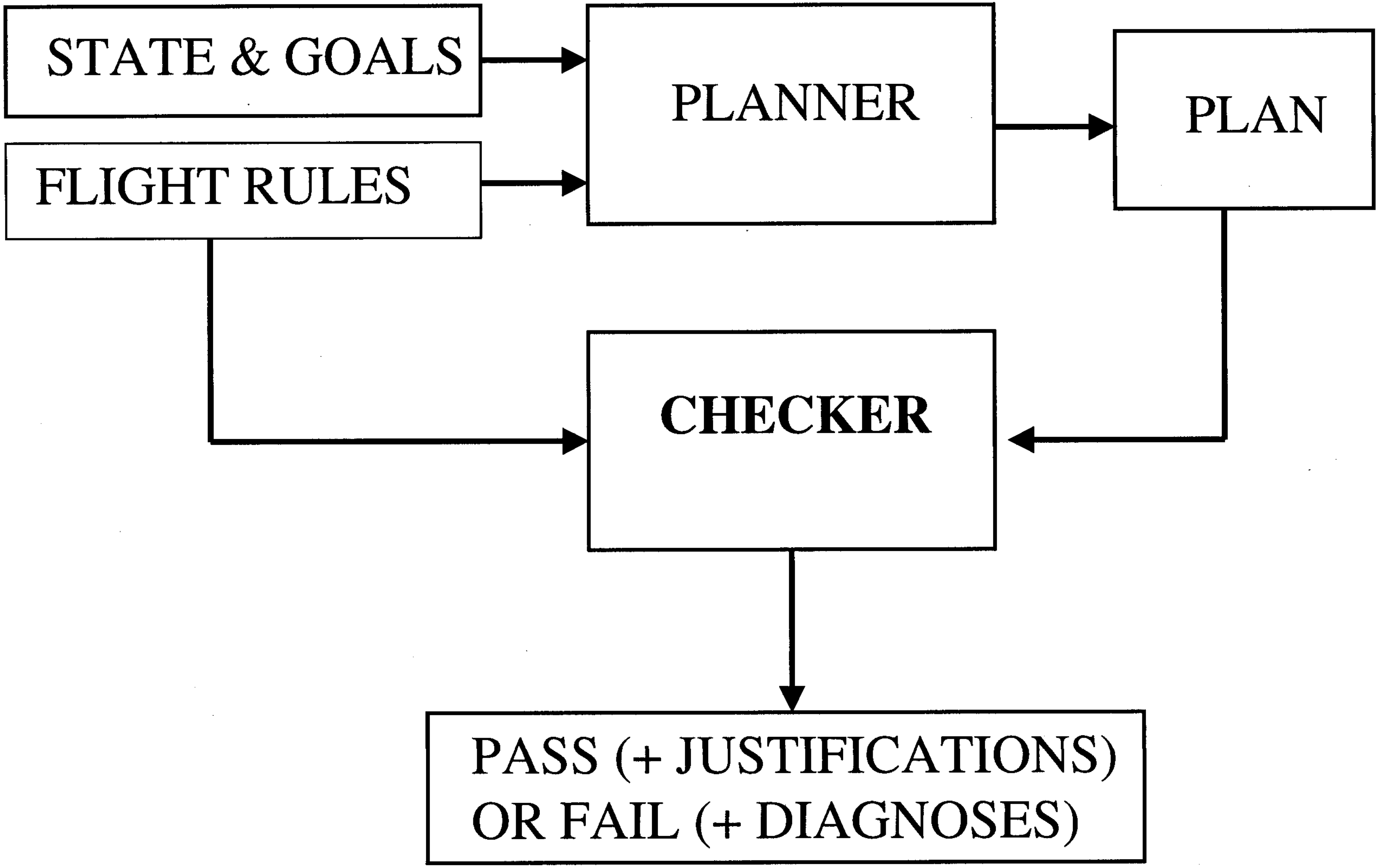
Rules involve time & parameters of states & activities

Small fragment of a plan

```
(#S(C-TOKEN
   :CARDINALITY :SINGLE    :NAME VAL-920
   :SV-SPEC (SPACECRAFT_ATTITUDE  SPACECRAFT_ATTITUDE_SV)
   :TYPE-SPEC ((CONSTANT_POINTING_ON_SUN
                    (HGA_AT_EARTH   BBC_DEADBAND_CRUISE)))
   :START-B-TOKEN VAL-920    :END-B-TOKEN VAL-920
   :STATE-VARIABLE (SPACECRAFT_ATTITUDE
                            SPACECRAFT_ATTITUDE_SV)
   :TOKEN-TYPE ((CONSTANT_POINTING_ON_SUN
                     (HGA_AT_EARTH    BBC_DEADBAND_CRUISE)))
   :DURATION (37801 500000000)
   :START-TIME-POINT TP-1279
   :END-TIME-POINT TP-1116
   :COMPAT-CONSTRAINTS ((CONTAINS 0 500000000 0 500000000)
                                    PS_WAYPT_1))    )
```

Designed to be read by software, not by humans!

Automate checking plans against flight rules



STATE & GOALS

FLIGHT RULES

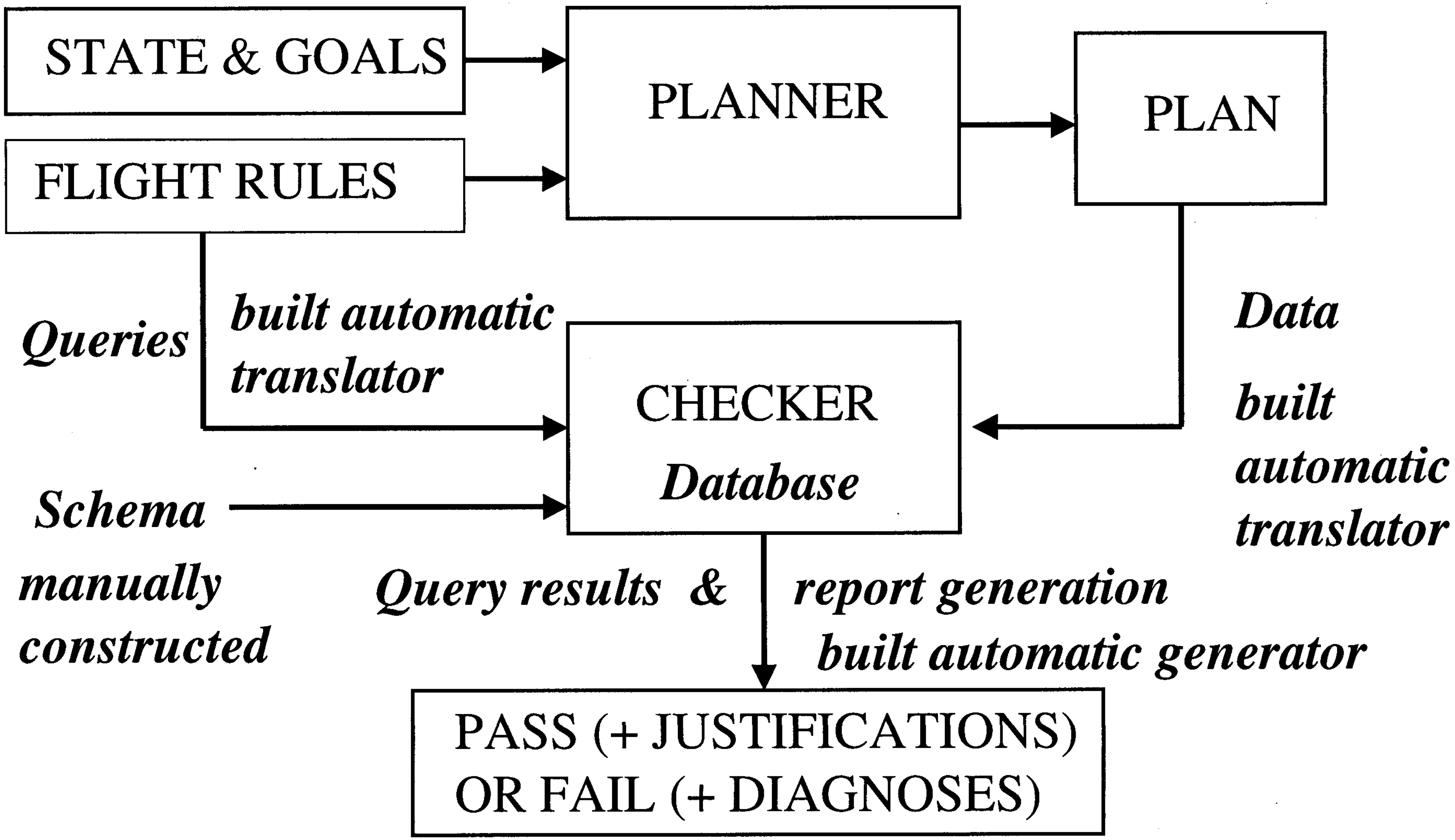
PLANNER

PLAN

CHECKER

PASS (+ JUSTIFICATIONS)
OR FAIL (+ DIAGNOSES)

Database used to perform checks



Metrics

- Automatically check every flight rule ***> 200***
- Applied to plans generated during testing ***thousands***
- Checking plans faster than generating plans
  30 seconds - 4 minutes < ***3 minutes - 10 minutes***
- Automatic regeneration of checker when flight rules change ***< 10 minutes; done 3 times***
- Development of checker lesser effort than of planner
  months < ***years***
- Accommodated a change to plan syntax ***< 3 days***

Redundancy & Rationale

A plan contains a sequence of activities and *justifications* of those activities -- *justification: activity* ←→ *flight rule(s)*

Rationale - planner arrives at the "right solution" (a plan that meets flight rules) for the "right reasons"

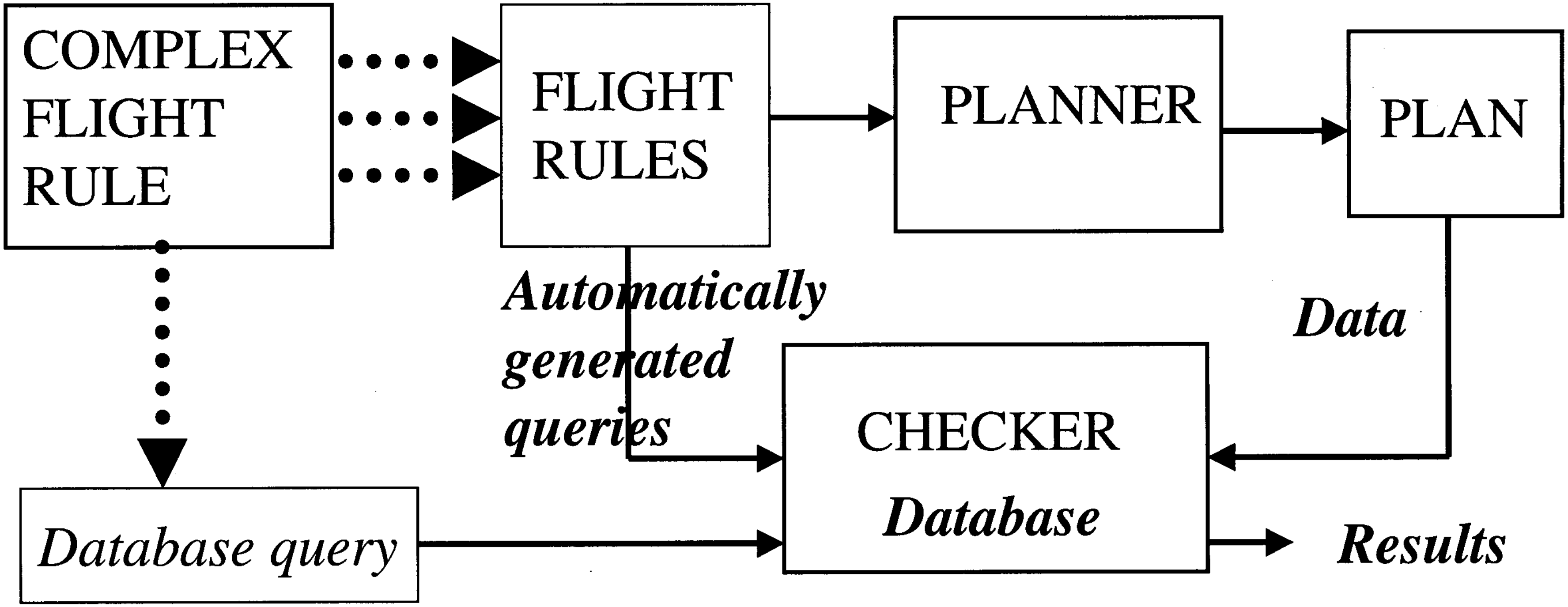
=> ***increased confidence in planner***

Redundancy - checker tests all the following:

- all activities of plan adhere to all flight rules
- each activity has a justification for every flight rule applicable to that activity
- every activity's justification can be traced back to an applicable flight rule

=> ***increased confidence in checker***

Validation



Complex flight rule *manually* decomposed and expressed as several planner flight rules.

Validate by manually expressing as a single database query.

Applicability - *worthwhile* when:

- Voluminous amounts of data to check
  - test run yields lots of data
  - lots of test runs
- Checking is difficult
  - many checks
  - checks hard to perform
- Cannot instead analyze the *generator* of the data
  - e.g., planner too complex for analysis via model checking

Applicability - *viable* when:

- Data self-contained w.r.t. check
- Data in machine-manipulatable form
  - e.g., plan is input to another program on spacecraft
- Check in machine-manipulatable form
  - e.g., flight rules expressed as planner constraints
- Checking easier than generation
  - e.g., planning - a core AI problem; checking easier
- Analysis language more expressive than requirements language
  - e.g., database language v.s. planner constraint language

Partnership development

Spacecraft planner expert - Ben Smith

Analysis expert - Martin Feather

- Spacecraft expert's time a critical resource
- Neither partner has time to become expert in both areas
- Analysis expert developed & maintained checker
- Spacecraft expert used checker
- Spacecraft expert extended checker (for validation)

Summary

- Autonomous systems raise challenges *and* opportunities
  - Challenges: many and detailed tests
  - Opportunities: test checking amenable to extensive automation
- Redundancy & Rationale increases confidence
  - Passed this test, but how much can we conclude?
  - Increased confidence in both planner and plan checker
- Validation
  - Gaps, where manually performed steps occur
  - Checking can bridge these gaps
- Partnership development
  - No one person can know and/or do everything
  - Spacecraft expert and analysis expert worked together